Towards an Integrated Vision across Inter-cooperative Grid Virtual Organizations

Ye Huang[1], Nik Bessis[2], Amos Brocco[1], Stelios Sotiriadis[2], Michele Courant[1], Pierre Kuonen[3], and Beat Hisbrunner[1]

[1] Department of Informatics, University of Fribourg, Switzerland
{ye.huang,amos.brocco,michele.courant,beat.hirsbrunner}@unifr.ch
[2] Department of Computing and Information Systems, University of Bedfordshire, UK
nik.bessis@beds.ac.uk, stelios@sotiriadis.gr
[3] Department of Information and Communication Technologies, University of Applied Sciences Western Switzerland
pierre.kuonen@hefr.ch

Abstract. Much work has been done to exploit the benefit brought by allowing job execution on distributed computational resources. Nodes are typically able to share jobs only within the same virtual organization, which is inherently bounded by various reasons such as the adopted information system or other agreed constraints. The problem raised by such limitation is thus related to finding a way to enable interoperation between nodes from different virtual organizations.

We introduce a novel technique for integrating visions from both resource users and providers, allowing to serve multiple virtual organizations as a whole. By means of snapshot data stored within each grid node, such as processing and interacting history, we propose a demand-centered heuristic scheduling approach named Critical Friend Community (CFC). To this end, a set of simplified community scheduling targeted algorithms and processing workflows are described. A prototype of our scheduling approach is being implemented within the SmartGRID project.

Keyword: Grid Scheduling, Meta-scheduling, Inter-cooperative, Critical Friend Community, SmartGRID, MaGate.

1 Motivation

During the last decade, the evolution of grid computing has resulted in different visions of the grid and technology that depend mainly on adopters point of view. A widely accepted vision is that homogeneous machines managed by a single Local Resource Management (LRM) system are interconnected as a cluster. Clusters connected within the same authorization constraint lead to a site, or grid node. Each site may have its own meta-scheduler, which behaves as a central manager, receiving job submission from grid users, and allocating approved jobs on the LRMs of local clusters for execution. Different sites can be organized into a Virtual Organization (VO) [1], according to reasons such as sharing the same Information System (IS) or geographical location.

Plenty of works have been done to enable effective and efficient job processing within the scope of cluster and single site [2] [3]. Although some research work [4] [5] [6] have started to exploit the incentive benefited from automatic cooperation amongst multiple sites (different meta-schedulers), there is still room for improvement in the area of enabling job exchange across different VOs in an automatic and self-manageable way. Collaboration between VO is typically made difficult because of different non-common factors, such as agreed constraint, resource discovery approach, geographical location, security, or user preference. Those factors could be so complicated and volatile that they cannot be expected to be understandable by other VOs. Therefore, realistic implementations normally assume that two nodes from different VOs, although they might be physically connected, are not aware of the existence of each other, despite the fact that they may have complementary job execution requirement and resource configuration. In this case, a notable issue has raised, namely how to facilitate the interaction and collaboration between complementary nodes, which are normally not aware of each other due to the boundaries of different VOs.

Our idea is to use heuristic data to facilitate the scheduling decision making process. Especially, exploiting historical interoperation metadata cached on each grid node would lead to a *demand centered* grid scheduling framework across multiple VOs.

2 Principle

As mentioned above, conventional grid VOs are bounded due to various non-common reasons, so that realistic job delegations only happen between nodes within the same VO. Such approach does not take the full advantage of the fact that a node could belong to more than one VO; especially while job delegation to a node of another VO will not broke the job submission constraints, e.g., critical security issue that only allows job execution within the same VO. In this case, the integrated vision of inter-cooperative VOs is named Critical Friend Community (CFC), which inherits from the pilot work of [7]. The idea is that knowledge of neighborhood grid topology and previous interactions (either directly or indirectly) are cached on each grid node, facilitating a more effective and efficient scheduling decision. The interconnected nodes known via historical realistic collaboration records are considered as Critical Friends (CF) to each other, and together they represent a Critical Friendship based Community that has crossed the boundaries of isolated VOs; furthermore, the strength of the Critical Friendship between nodes of CFC is determined by more “subjective and empirical” factors, such as the quantity and quality of previous interactions.

3 Approach

In order to achieve a Critical Friend Community to enable interaction and collaboration between nodes of different VOs, a set of conceptions need to be detailed, which include:

– A topology used to interconnect nodes from separated VOs.
– A scheduling model that is able to make decisions based on information from both infrastructure providers, and knowledge of Critical Friends.
– A way of organizing and presenting knowledge of Critical Friends on each node.

3.1 Topology

Regarding the definition and classification of grid computing [8], a noteworthy point is that grids are characterized and categorized by their hardware and software properties, including operating system, machine architecture, security compromise, etc. In other words, the grids are characterized according to a provisioning perspective, and are thus *provision centered*.

One of the tendencies illustrated by emerging technologies, such as virtualization and cloud computing [9], has demonstrated that novel approaches that can remedy user's burden of deciding where to submit his jobs have a promising future. The grid services responsible for mapping jobs on proper resources are supposed to execute well-presented jobs automatically as long as appropriate resources exist and are physically interconnected, regardless of the affiliated VOs. In contrast to the traditional grid characteristics, the novel approach can thus be considered as *demand centered*. The *demand centered* topology represents a decoupled network focused on submitted job requirement, which is fundamentally different from the *provision centered* based vision.

Our idea of implementing a *demand centered* topology is the design of the Critical Friend Community (CFC). As illustrated in Figure 1, a CFC is comprised of nodes (in our case, a node refers to a grid site) affiliated to different VOs, thus the interaction between CFC nodes may cross the borders between multiple VOs. Each CFC node has a snapshot used to store historical interaction records, as well as the profile of contacted remote nodes. Each contacted remote node in the snapshot is considered as a Critical Friend of the owner node, and the Critical Friendship is weighted by the quantity and quality of previous collaboration records.

Once a node notifies a local job submission (job submitted by the user of the same site), it is considered as a job request initiator, and is able to process the following solutions to handle the job request, either sequentially or simultaneously:

– Solution 1: The initiator allocates the job to a locally owned LRM for local execution.
– Solution 2: The initiator searches for appropriate remote nodes within the same VO using the adopted Information System, and delegates the job to a discovered remote node.

– Solution 3: The initiator selects one of its Critical Friends (a remote node either from the same VO or not), depending on relevant parameters such as job profile, friendship category, friendship weight, remote nodes community profile and recent status, and delegates the job to such Critical Friend.

The first solution matches the typical scenario of the a traditional grid, where the meta-scheduler only interacts with LRMs on which it has complete knowledge and full control. The second solution broads the view by invoking nodes within the same VO, thus requiring a commonly shared Information System to guarantee that nodes of the same VO are able to find/interact with each other. The third solution behaves in a similar way with solution 2. However, solution 3 selects candidate nodes from the initiator node's Critical Friends, which are determined by means of previous interaction records independently from all constraints imposed by VO boundaries. If a selected Critical Friend is not capable of disposing the job delegation request, it could pass such request to its own friends if allowed by the terms agreed with the request initiator node. Because Critical Friends of a single node may be affiliated to different VOs, a job request can be therefore transferred within the scope of the CFC, while inconsistencies caused by VO boundaries are filled by agreed terms represented via the Critical Friend Relationship (CFR). In other words, solution 3 emphasizes on maintaining a Critical Friends Community (CFC), which is issued by *demand centered* collaboration experience, instead of factors introduced by *provision centered* grid infrastructure.

The concept of CFC mirrors the notion of relationships occurring in the real world. If a person (node in our case) is looking for a specific service but neither owns one nor knows where to get it, he will ask some of his friends (Critical Friends) who used to be helpful (decision based on past experience). If they are not able to do the favor, these friends will pass the request to their friends hoping that someone across the knowledge group (virtual organization in our case) will have the expected capability. Based on information given back via friends network, the original requester could make a decision, and invoke the service provided by friends (direct or indirect) if necessary, under agreed terms.

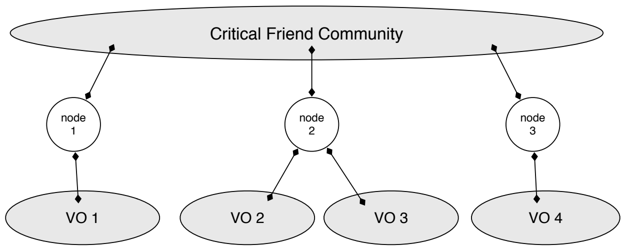

Fig. 1. Critical Friend Community Topology

3.2 Community Scheduling Model

In our work, the scheduling model is able to make decisions based on information from both infrastructure providers and knowledge of Critical Friends. We call it the Community Scheduling Model (CSM). The CSM is carried out by a coordinator component. Taking into consideration that each node within the CFC has its own local scheduling policies, as well as full control of the local resources, the coordinator is supposed to collaborate with the existing local scheduling policies, and provide a broad view by enabling the participation of remote nodes.

A coordinator is different from a meta-scheduler, although both could be physically the same component sometimes. A meta-scheduler simply assigns a job to local LRM for execution, while two coordinators have to negotiate on a job delegation from one node to the other. That is to say that a job delegation request issued by a coordinator can be refused or altered, which is not the case for a meta-scheduler.

If a job delegation request is refused or altered, the initiators coordinator has to continue by either reporting the failure to user, or releasing a re-negotiation process with modified parameters. The approach to automate the above process can be comprehended as a *workflow based* schedule, because the coordinator has already determined steps to do for handling subsequent behaviors like re-negotiation and failure. Regarding the scheduling process of each CFC node concerns many volatile factors retrieved from various environments, adaptability is a critical capability for the Community Scheduling Model, in order to exploit potential opportunities of fulfilling received job execution requests without bothering the initiator user.

Currently, the Community Scheduling Model and a set of detailed algorithms are under development. More specifically, implementation includes:

Job Orchestrating Algorithm (JOA). The philosophy of JOA is to organize a to-process job queue by merging diverse job incoming sources, with respect to local user preference.

If the preference indicates that local jobs have higher priority, the JOA will try to fill the size limited output queue with jobs from local queue firstly, and pick appropriate jobs from other sources, e.g., community queue or unprocessed queue, only if the limit of the output queue is not exceeded. If the user desires an equal treatment for all incoming job requests, the output queue will be comprised of the earliest arrived jobs, no matter where they come from. Finally, if a profitable philosophy is determined, each arrived job will be evaluated, in order to determine individual *job-profite-rate* value. In this case, the output queue will be composed by the most profitable jobs. Once the output queue is generated, the local policy of the participating node is invoked for future processing.

Furthermore, the JOA can be extended by users self-defined job orchestrating policies, and other locally adopted scheduling algorithms, besides herein mentioned FCFS and EasyBackfilling.

Resource Orchestrating Algorithm (ROA). The ROA is responsible for generating a set of appropriate candidate resources for each input job request, depending on user preference.

If the *LocalResourcePriority* policy is chosen, resources owned by the local node are considered firstly, with an additional selection from other list (e.g., community resource list and critical friend resource list) only occurring if the size limit of the output resource list is not achieved. If the policy *CommunityResourceFair* is preferred, a fair selection is carried out on all known resource list. Finally, if the policy *FriendResourcePrioriy* is specified, the output list will firstly pick up a suitable resource owned either locally or by some critical friends, with other list not being considered unless the output resource list is not full.

Similarly to the aforementioned JOA, the ROA can be extended by user self-defined resource orchestration policies, but only if the expected known resource list can be found within the local node's snapshot storage.

Community Scheduling Algorithm (CSA). Once a candidate schedule (a job with its candidate resource list) arrives, an allowed maximum scheduling time duration will be given to prevent unacceptable delays and performance loss. The CSA is responsible for contacting the candidate resources simultaneously within allowed delay, in order to get a job allocation/delegation *agreement* based on the expected request (in our case, it is an *agreement offer*). An *agreement* means that the job execution request is approved by the target resource (either locally or remotely) and if such job can be delivered within a certain time, it will be accepted and executed under the agreed terms. As soon as an *agreement* has been made between the requesting node and a target resource, other *agreement offers* will be revoked.

In case no candidate resources are able to accept such *agreement offer* due to various reasons, e.g., local workload, local policy alternation, latest resource status change, the CSA needs to check whether the allocated scheduling time has expired. If not, the CSA is able to contact the locally adopted Information System, and asks for a live search from the located VO within the remaining scheduling duration. If appropriate resources can be found within such time constraints, a parallel (re-)negotiation with a newly prepared *agreement offer* can be issued again, within the shortened time duration.

As mentioned, although the job allocation is a different operation from job delegation (because the targeted resource of job allocation is an owned LRM of the local node, which cannot negotiate a job acceptance), the CSA doesn't concern such slight difference by ignoring the *agreement offer* based (re-)negotiation process if the target resource is managed by a local LRM.

3.3 Metadata Snapshot

Designing a snapshot based decentralized data warehouse strategy concerns several crucial considerations, including: storage policy, data scheme, and information exchange model.

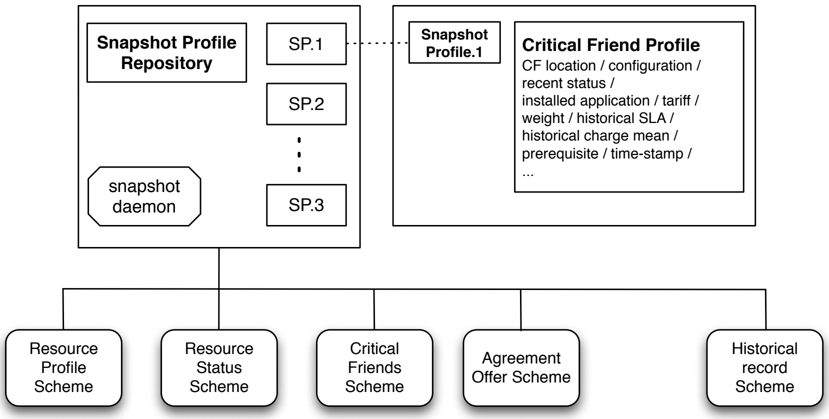
Snapshot Profile Repository
SP.1
SP.2
SP.3
snapshot daemon
Snapshot Profile.1
Critical Friend Profile
CF location / configuration / recent status / installed application / tariff / weight / historical SLA / historical charge mean / prerequisite / time-stamp / ...
Resource Profile Scheme
Resource Status Scheme
Critical Friends Scheme
Agreement Offer Scheme
Historical record Scheme

Fig. 2. Metadata Snapshot Structure

The metadata stand for information collected during each CFC node's processing history. Different kind of metadata can be collected by different adopted utilities. For instance, resource characteristics are provisioned by either a dedicated monitoring system, or an applicable grid scheduler; similarly, Critical Friend interoperation knowledge is monitored by a specific service. Finally, a weighting system provides advanced information by calculating and optimizing obtained data mentioned above.

Regarding many factors that could impact the scheduling decision, each CFC node has a metadata snapshot to preserve data provisioned for its CSM. Each metadata snapshot is comprised of sets of schemes. A scheme is a group of elements that is used for describing a particular resource or purpose. For example, a machine scheme is normally composed of elements such as machine architecture, operating system, number of CPU, etc. Other important schemes include: local resource profile, local resource status, agreement offer list, known CF (Critical Friend) profile list, known CF recent status list, historical processing records, etc. Noteworthy, data stored within each scheme is kept up-to-date over time, and is being evaluated and weighted to facilitate intelligent scheduling for the future incoming job requests.

As illustrated in Figure 2, different schemes are used together to construct a Snapshot Profile (SP) by the means of a *Snapshot Daemon*. A Snapshot Profile is a group of information encoded in a machine-processable schema using a markup language, and represents certain kind of capability provided by a Critical Friend. The Snapshot Profile decouples the implementation between job scheduling and metadata collecting, and can be used for searching competent Critical Friends by node scheduler/coordinator.

To remedy the pain of organizing all necessary information (static and dynamic) of a Critical Friend, as well as to represent such knowledge is easy-to-understand way, the notion of *Snapshot Profile* is proposed. A *Snapshot Daemon* is responsible for gathering metadata distributed in different schemes, and representing the knowledge of each individual Critical Friend (CF) in a clean and well-organized way. The Snapshot Profile concerns all valuable knowledge of a Critical Friend, including: CF location, configuration, static and dynamic status, installed application list, tariff, weight (as a CF of the local node), historical SLA (depending on job type), historical charge-load arrange (depending on job type), prerequisite (depending on job type), time-stamp (indicating until when this information can be considered as up-to-date).

Finally, the *Snapshot Daemon* is also responsible of handling metadata exchange, either proactively or reactively, with other CFC nodes.

4 Conclusion and Future Work

This paper presents the concept of Critical Friend Community (CFC), which is inspired by the motivation of enabling interoperation between nodes from isolated grid virtual organizations. Regarding the real grid virtual organizations, which are normally bounded due to *provision centered* factors, the notion of Critical Friend Community is raised from a *demand centered* prospect. The kernel idea is that previous interaction experience, or "partner trust", overweights the physical boundaries. With this in mind, nodes of CFC are supposed to take advantage of historical data retained by each node, such as information exchange and job delegation records, to construct collaboration across multiple virtual organizations. It is noteworthy that the strength of relationship between participating nodes, i.e. the Critical Friendship, is determined by the number of previous interactions, as well as their quality.

To achieve the goals of the CFC, a Community Scheduling Model (CSM) is introduced, which respects the reality that each participating grid node has its own local scheduling polices. The CSM provides a broad view by allowing job exchange between local node and remote node by the mean of negotiation.

Regarding many volatile factors could impact the decision made by the CSM, a metadata snapshot design is proposed. This allows to assemble data collected from diverse sources to build a standard profile depending on local node's preference. Such design decouples the implementation between job scheduling and metadata collection, and matches the philosophy of CFC, i.e. a flexible approach to achieve *demand centered* prospect.

Current research focuses on a coordinator prototype, which is under implementation based on existing MaGate scheduler from the SmartGRID project that provides a platform independent communication infrastructure between nodes. Furthermore, the CSF based community scheduling processing components, such as the simplified Job Orchestrating Algorithm (JOA), Resource Orchestrating Algorithm (ROA), and Community Scheduling Algorithm (CSA), are being implemented within the aforementioned coordinator. Finally, information collected by our currently adopted Information System, such as [10], will be translated into corresponding metadata snapshot schemes, in order to provide an easy-to-use semantic knowledge during the community scheduling process.

5 Acknowledgements

Critical Friend Community (CFC) is an extended work on the ongoing SmartGRID project[1,2]. This work is supported by the Swiss Hasler Foundation[3], in the framework of the ManCom Initiative (ManCom for Managing Complexity of Information and Communication Systems), project Nr. 2122.

References

1. Foster, I., Kesselman, C., Tuecke, S.: The Anatomy of the Grid: Enabling Scalable Virtual Organizations. International Journal of High Performance Computing Applications 15(3), 200 (2001)
2. Krauter, K., Buyya, R., Maheswaran, M.: A taxonomy and survey of grid resource management systems for distributed computing. Software: Practice and Experience (2002)
3. Schopf, J.: Ten actions when superscheduling: A grid scheduling architecture. In: Workshop on Scheduling Architecture, Global Grid Forum, Tokyo (2003)
4. Yarmolenko, V., Sakellariou, R.: Towards increased expressiveness in service level agreements. Concurrency and Computation: Practice and Experience 19(14) (2007)
5. Waldrich, O., Wieder, P., Ziegler, W.: A meta-scheduling service for co-allocating arbitrary types of resources. In: Wyrzykowski, R., Dongarra, J., Meyer, N., Waśniewski, J. (eds.) PPAM 2005. LNCS, vol. 3911, pp. 782–791. Springer, Heidelberg (2006)
6. Huedo, E., Montero, R., Llorente, I.: The GridWay framework for adaptive scheduling and execution on grids. Scalable Computing: Practice and Experience 6(3), 1–8 (2005)
7. Huang, Y., Bessis, N., Brocco, A., Kuonen, P., Courant, M., Hirsbrunner, B.: Using Metadata Snapshots for Extending Ant-based Resource Discovery Service in Intercooperative Grid Communities. In: International Conference on Evolving Internet, INTERNET 2009, Cannes, French Riviera, France. IEEE Computer Society, Los Alamitos (2009)
8. Foster, I., Kesselman, C.: The grid: blueprint for a new computing infrastructure. Morgan Kaufmann, San Francisco (2004)
9. Buyya, R., Yeo, C., Venugopal, S., Broberg, J., Brandic, I.: Cloud computing and emerging IT platforms: Vision, hype, and reality for delivering computing as the 5th utility. Future Generation Computer Systems 25(6), 599–616 (2009)
10. Brocco, A., Frapolli, F., Hirsbrunner, B.: Bounded diameter overlay construction: A self organized approach. In: IEEE Swarm Intelligence Symposium, SIS 2009. IEEE, Los Alamitos (2009)

[1] http://diuf.unifr.ch/pai/smartgrid
[2] http://gridgroup.hefr.ch/smartgrid
[3] http://www.haslerstiftung.ch/